ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization *Vol.5, Special Issue 5, June 2017*

8th One Day National Conference on Innovation and Research in Information Technology (IRIT- 2017)

Organized by

Departments of ISE, CSE & MCA, New Horizon College of Engineering, Bengaluru, Karnataka 560103, India

A Survey on MINIX 3 OS with Dynamic Timeout Choice

Sridevi.S[1], Sai Bhargavi.P[1], Prof Dr.Vishwanath .Y[2]

BE Student, Dept. of ISE, New Horizon College of Engineering, Bangalore, Karnataka, India [1]

Sr. Asst. Professor, Dept. of ISE, New Horizon College of Engineering, Bangalore, Karnataka, India [2]

ABSTRACT: Most of the Operating system with monolithic kernel design often failed due to the presence of bugs in device drivers. Thus microkernel design is introduced in which the servers and drivers are implemented in the user mode by minimizing the OS kernel. Minimizing the OS kernel helps in reducing the damage caused by bugs in the code. A relevant issue in microkernel architecture is unresponsive components due to infinite loops and deadlocks. For detecting the unresponsive components in the microkernel a proposed strategy is heartbeat mechanism ie., a monitor sends heartbeat messages to a component which should reply within a timeout. The choice of timeout is tricky as it should be dynamically adapted to the load conditions of the system. Thus an adaptive heartbeat mechanism is considered in which the timeout is estimated from the previous response times. The timeout choice is made by using estimation algorithm in the context of Minix 3 OS.

KEYWORDS: Microkernel; Minix 3 OS; Heartbeat mechanism; Unresponsive components;

I. INTRODUCTION

Operating system is the most fundamental system program. The work of the OS is to control all the computer's resources and it provides a platform on which application programs can be written. Large monolithic kernels where used to implement early operating systems. All the operating system functionality runs in kernel mode in the monolithic kernel design, due to which any bug can crash the entire system. Thus the monolithic kernel design lacks proper fault isolation. Scheduling, file system, networking, device drivers, memory management, paging, and more – the complete operating system was packed to form a single kernel. Implementing the servers outside the kernel and reducing the kernel can be done by the microkernel approach. The microkernel MINIX 3 is POSIX compliant operating system which is designed to be highly reliable, secure and flexible. By breaking the system into number of self contained modules, the microkernel approach brings the ideas of fault isolation and modularity. The microkernel architecture comprises of most of the unprivileged components that implements several services and a small privileged component(microkernel) of limited number of lines of code that implements basic services. The microkernel architecture can be advantageous to deal with unresponsive components. Components that are not crashed but do not reply to any of the service request made are called unresponsive. This is a common issue and is caused by software faults such as infinite loops, software aging and deadlocks.

II. RELATED WORK

1) Reorganizing UNIX for Reliability

Operating systems are expected to function flawlessly, but, unfortunately, most of today's operating systems fail all too often. These failures are mainly due to the underlying monolithic kernel design. As believed, reducing the OS kernel and running drivers and otherr core components in unpriviledged user mode helps to minimize the damage that may be caused by bugs in such code. Therefore, the entire operating system is run as a collection of independent, tightly restricted, user-mode processes. This structure is also combined with several explicit mechanisms for transparent recovery from crashes and other failure and hence resulting in highly reliable,

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization Vol.5, Special Issue 5, June 2017

8th One Day National Conference on Innovation and Research in Information Technology (IRIT- 2017)

Organized by

Departments of ISE, CSE & MCA, New Horizon College of Engineering, Bengaluru, Karnataka 560103, India

multiserver OS that still looks and feels like UNIX, with performance loss of only 5% to 10% and is called MINIX 3.

2) Towards Real Microkernel

Large monolithic kernels where used to implement early operating systems. All the operating system functionality runs in kernel mode in the monolithic kernel design, due to which any bug can crash the entire system. Thus the monolithic kernel design lacks proper fault isolation. Scheduling, file system, networking, device drivers, memory management, paging, and more – the complete operating system was packed to form a single kernel. Implementing the servers outside the kernel and reducing the kernel can be done by the microkernel approach.

Most older microkernel evolved from monolithic kernels are though, theoretically advantageous, they did not achieve sufficient flexibility and performance. Applying performance criterion to such a complex system is not trivial. Naive, uninterpreted measurements are sometimes misleading. Most problems of the first generation microkernel were caused by their step by step development.

3) Failure Resilience for Device Drivers

Most of the operating system crashes are due to bugs in the device drivers than in any other code. The architecture of failure resilient OS is presented which can help recovery from dead device drivers and other components with the help of post mortem recovery.

This operating system consists of multiple isolated user-mode components which are structured in a way that the system can automatically detect and repair a broad range of defects without user intervention. The Reincarnation server manages all system processes and constantly monitors the system's health. The data store provides naming services and can be used to recover lost state after a crash. When a failure is detected by defect detection mechanism, the defect is repaired by policy-driven recovery procedure and the system continues to run all the time with minimum disturbance to other processes. Malfunctioning components may be replaced in a policy-driven recovery procedure to masks failures for both users and applications. The measured performance overhead due to the recovery mechanisms can be as low as 1%. Thus, working on failure resilience for device drivers represents a small step towards more dependable operating systems.

4) Fault Isolation for Device Drivers

This work explores the principle and practice of isolating low-level device in order to improve OS dependability. Extensions such as device drivers are responsible for the majority of OS crashes. The reason that these crashes can occur is the close integration of (untrusted) extensions with the trusted core kernel and memory corruption was found to be one of the main OS crash causes.

Fixing buggy drivers is infeasible since configurations are continuously changing with, while there is a consensus that drivers need to be isolated this issue is addressed by principle of least authority.

MINIX 3 restricts drivers based on the principle of least authority in order to limit the damage that can result from bugs. Fault isolation is achieved through a combination of structural constraints imposed by a Multi-server design and run-time memory granting. In general many of these techniques are ported and applicable to other systems.

5) Adaptive Monitoring in Microkernel OS

A relevant issue of microkernel architecture is unresponsive components because of deadlocks and infinite loops. The strategy used to detect unresponsive components is Heartbeat mechanism ie, A monitor sends a heartbeat message to unresponsive components and these components should reply within the timeout choice. Here, an adaptive heartbeat mechanism is used for which timeout choice is estimated from past response times. In today's complexity in OS, it is very common to have software faults. Therefore to reduce the faults, Microkernel OS are used for developing more reliable systems. In particular to this paper, this architecture can be exploited to deal with unresponsive components, i.e the component is not crashed but it does not reply to any service request

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization *Vol.5, Special Issue 5, June 2017*

8th One Day National Conference on Innovation and Research in Information Technology (IRIT- 2017)

Organized by

Departments of ISE, CSE & MCA, New Horizon College of Engineering, Bengaluru, Karnataka 560103, India

A proposed strategy for detecting unresponsive components in microkernel is heartbeat mechanism, ie "Ping " message is sent to all components and if the reply is not received by any component within the timeout choice, then that component is detected as unresponsive component.

Two different scenarios can be considered when components are pinged :

(i) If the timeout is too short, a component that is performing a time-consuming computation can be considered as unresponsive.

(ii) If the timeout is too long, component is not detected as unresponsive, even if it is.

To overcome the above two scenarios, this paper adaptively selects the timeout choice that is implemented in MINIX 3 microkernel OS. The dynamic timeout can be computed by using either of the following algorithm :

(i) EWMA (Exponential Weighted Moving Average) algorithm

(ii) Weighted Sum Algorithm

(iii) Max Algorithm

By evaluating all the three algorithms, Weighted Sum or Max algorithm is considered as efficient when little false positives and latency is needed.

Future work encompasses the possibility to evaluate the algorithms in different Scenarios

(i) Considering more random factors such as I/O and Memory Management.

(ii) Scalability Approach : By increasing the number of monitored servers.

III. ANALYSIS

Thus we have studied 4 different types of TV program recommendation techniques. In future we will work to construct a more precise and efficient recommendation technique on different fields.

IV. CONCLUSION

We have seen the success of recommender system in E-commerce, the same idea can be applied to the TV program recommendation. Only challenge is to make a platform where users can share their profile and provide feedback, similar to E-commerce websites. In this literature review (LR) we have classified and analyzed 4 different recommendation systems. There are lots of recommendation techniques available. They are costly and difficult to analyze the data sometimes, chances are less for viewers to like all the recommended programs, the prediction is based on probability only. Still the algorithms can be improved by using good statistics and efficient use of the data.

REFERENCES

[1] Jorrit N. Herder, Herbert Bos, Ben Gras, Philip Homburg, and Andrew S. Tanenbaum Reorganizing UNIX for Reliability Computer Science Dept., Vrije Universiteit Amsterdam De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

[2] Jochen Liedtke, " Towards Real Microkernel", September 1996

[3] J. N. Herder et al., "Failure Resilience for Device Drivers", in Conf. on Dependable Systems and Networks, 2007.

[4] Jorrit N. Herder, Herbert Bos, Ben Gras, Philip Homburg, and Andrew S. Tanenbaum "Fault Isolation for Device Drivers" Dept. of Computer Science, VU University Amsterdam, The Netherlands 2009 IEEE

[5] Domenico Cotroneo, Domenico Di Leo, Roberto Natella "Adaptive monitoring in microkernel Oss" 2010 International Conference on Dependable Systems and Networks Workshops

[6] Su Myeong Lee and Hee Yong SungKyunKwan University, "Dynamic Window-based Adaptive Fault Monitoring for Ubiquitous Computing".

[7] E. Weyuker, "Testing Component-Based Software: A Cautionary Tale," 1998.

[8] M. Sullivan and R. Chillarege, "Software Defects and their Impact on System Availability," 1991

[9] Domenico Cotroneo , Roberto Natella, and Stefano Russo, "Assessment and Improvement of Hang Detection in the Linux Operating System"

[10] Jorrit N. Herder, Herbert Bos, Ben Gras, Philip Homburg, and Andrew S. Tanenbaum "Construction of a Highly Dependable Operating System".

[11] Jorrit N. Herder, Herbert Bos, Ben Gras, Philip Homburg, and Andrew S. Tanenbaum "Load Balance Facility in Distributed MINIX System", National Cheng-Kung University Tainan , Taiwan, ROC

[12] Jorrit N. Herder, David C. van Moolenbroek, Raja Appuswamy, Bingzheng Wu, Ben Gras, and Andrew S. Tanenbaum "Dealing with Driver Failures in the Storage Stack".

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization *Vol.5, Special Issue 5, June 2017*

8th One Day National Conference on Innovation and Research in Information Technology (IRIT- 2017)

Organized by

Departments of ISE, CSE & MCA, New Horizon College of Engineering, Bengaluru, Karnataka 560103, India

[13] Yennun Huang and Chandra Kintala Nick Kolettis and N. Dudley F'ulton "Software Rejuvenation: Analysis, Module and Applications".

[14] Long Wang, Student Member, IEEE, Zbigniew Kalbarczyk, Member, IEEE, Weining Gu, Member, IEEE, and Ravishankar K. Iyer, Fellow, IEEE "Reliability MicroKernel: Providing Application-Aware Reliability in the OS".

[15] " Operating Systems Design and Implementation", Third Edition By Andrew S. Tanenbaum - Vrije Universiteit Amsterdam, The Netherlands, Albert S. Woodhull - Amherst, Massachusetts.